# AppleScript and Cocoa Bridge

## Session 403

These are confidential sessions—please refrain from streaming, blogging, or taking pictures.

At the convention center, a movie will be placed here

# AppleScript and Cocoa Bridge

**Christopher Nebel**
Neb-El, Last Son of Krypton

# Introduction

- Cocoa is *the* framework for Mac OS X applications
- Originally for Objective-C…
- …but other languages work too, via Cocoa bridges
  - Python
  - Ruby
  - AppleScript

# What You'll Learn

- What's new in AppleScript
- How to write a Cocoa application using AppleScript

# AppleScript

# What's New in Snow Leopard?

- Developer improvements
- Language improvements
- Tool improvements
  - Better syntax coloring
  - Better logging
  - Multithreading

# Demo

## AppleScript

**Michael Gorbach**
Just like “Gorbachev,” but without the “ev”

# AppleScript in Snow Leopard

## Tool improvements

- AppleScript Editor
  - /Applications/AppleScript/Script Editor is now /Applications/Utilities/AppleScript Editor
  - Multithreaded
  - Improved event logging
  - Finer-grained syntax coloring

# AppleScript in Snow Leopard

## Language improvements

- A lot of bug fixes
  Radar 1186965: Only the first element of “text item delimiters” takes effect
  Radar 1402339: Make AppleScript thread-safe
  Radar 2206026: Getting elements of strings is slow
- Performance improvements
- No dependency on Window Server
- Improved unicode and locale support
- See AppleScript release notes for details

# AppleScript in Snow Leopard

## Developer improvements

- 64-bit
- Thread-safe
  - Works on background threads
  - Works on multiple threads
- Scripting additions may not be background thread-safe
  - Addition will execute on main thread
  - Additions should add `ThreadSafe` to Info.plist (see TN1164)

# AppleScript in Snow Leopard

## Developer improvements

- Improved security for scripting additions
- For scripters
  - Move addition commands out of `tell` blocks or use `tell me to`
  - See AppleScript release notes
- For addition developers
  - Add `Context` information to Info.plist
  - See TN1164

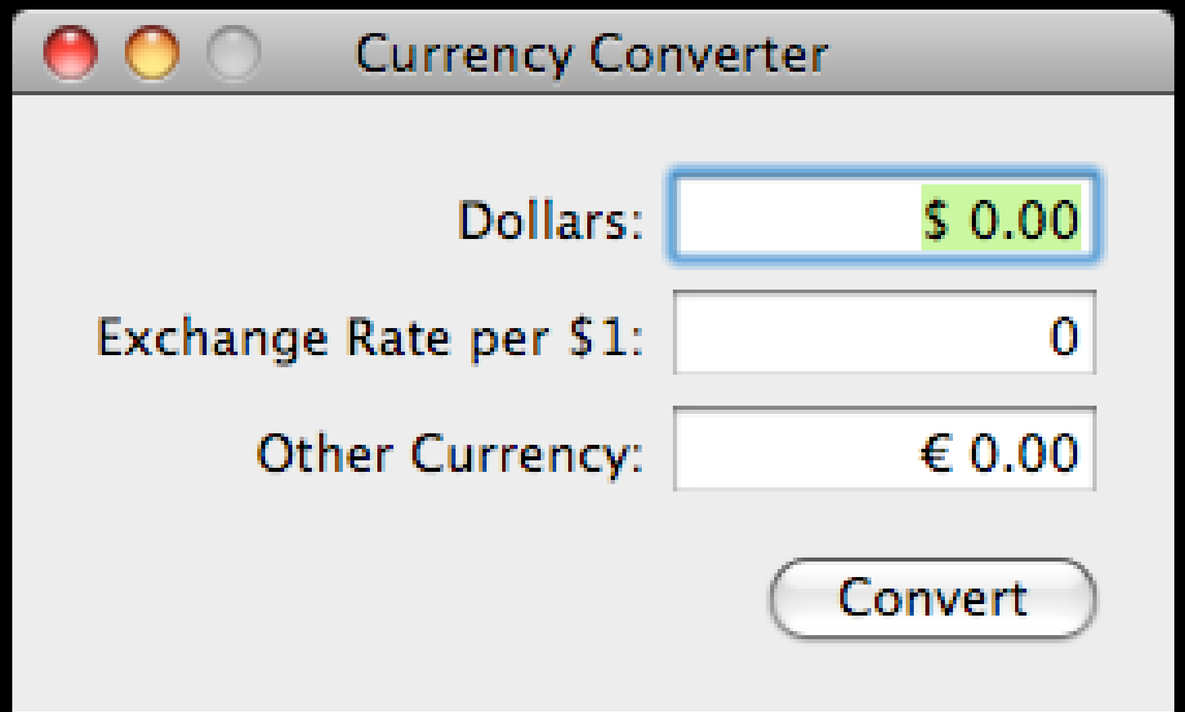
# AppleScript in Snow Leopard

- Many refinements
- Improved tools
- Future-compatible
  - 64-bit
  - Thread-safe

# AppleScript and Cocoa

# Compare and Contrast

## Currency Converter



# AppleScript Studio

## Currency Converter

```
on clicked sender
   tell window of sender
      set amt to contents of text field "amount"
      set rate to contents of text field "rate"
      set total to converter's convertAmount(amt, rate)
      set contents of text field "total" to total
   end tell
end clicked
```

# AppleScript Studio

## Currency Converter

```
on clicked sender
   tell window of sender
      set amt to contents of text field "amount"
      set rate to contents of text field "rate"
      set total to converter's convertAmount(amt, rate)
      set contents of text field "total" to total
   end tell
end clicked
```

# AppleScript Studio

## Currency Converter

```
on clicked sender
   tell window of sender
      set amt to contents of text field "amount"
      set rate to contents of text field "rate"
      set total to converter's convertAmount(amt, rate)
      set contents of text field "total" to total
   end tell
end clicked
```

# Objective-C

## Currency Converter

```
@interface ConverterController : NSObject {
    IBOutlet NSTextField *amountField;
    IBOutlet NSTextField *rateField;
    IBOutlet NSTextField *totalField;
}
@end


@implementation ConverterController
- (IBAction) convert: (id) sender {
    float amount = [amountField floatValue];
    float rate = [rateField floatValue];
    float total = [converter convertAmount:amount atRate:rate];
    [totalField setFloatValue:total];
}
@end
```

# Objective-C

## Currency Converter

```
@interface ConverterController : NSObject {
    IBOutlet NSTextField *amountField;
    IBOutlet NSTextField *rateField;
    IBOutlet NSTextField *totalField;
}
@end


@implementation ConverterController
- (IBAction) convert: (id) sender {
    float amount = [amountField floatValue];
    float rate = [rateField floatValue];
    float total = [converter convertAmount:amount atRate:rate];
    [totalField setFloatValue:total];
}
@end
```

# Objective-C

## Currency Converter

```
@interface ConverterController : NSObject {
    IBOutlet NSTextField *amountField;
    IBOutlet NSTextField *rateField;
    IBOutlet NSTextField *totalField;
}
@end


@implementation ConverterController
- (IBAction) convert: (id) sender {
    float amount = [amountField floatValue];
    float rate = [rateField floatValue];
    float total = [converter convertAmount:amount atRate:rate];
    [totalField setFloatValue:total];
}
@end
```

# Objective-C

## Currency Converter

```
@interface ConverterController : NSObject {
    IBOutlet NSTextField *amountField;
    IBOutlet NSTextField *rateField;
    IBOutlet NSTextField *totalField;
}
@end


@implementation ConverterController
- (IBAction) convert: (id) sender {
    float amount = [amountField floatValue];
    float rate = [rateField floatValue];
    float total = [converter convertAmount:amount atRate:rate];
    [totalField setFloatValue:total];
}
@end
```

# AppleScript and Cocoa

## One of these things is not like the others

- AppleScript Studio ≠ Cocoa
  - Can’t use standard Cocoa tools
  - Can’t use standard Cocoa documentation
  - Can’t use standard Cocoa community
  - Can’t use a lot of Cocoa frameworks
- Goal: Make AppleScript-Cocoa development work like the rest

# AppleScript Studio

## I come not to praise AppleScript Studio…

- Deprecated as of Snow Leopard
  - Deployed applications still work
  - Can continue to develop existing projects
- Development features hidden

```
defaults write com.apple.InterfaceBuilder3
   IBEnableAppleScriptStudioSupport -bool YES
```

# AppleScriptObjC

## Cocoa bridge for AppleScript

- Built into Snow Leopard
- Cocoa bridge for other languages
  - Tools
  - Documentation
  - Community
  - Any Objective-C API
- Brings AppleScript in line with other languages
  - Normal Cocoa techniques just work!

# AppleScriptObjC

## Currency Converter

```
@interface ConverterController : NSObject {


    IBOutlet NSTextField *amountField;
    IBOutlet NSTextField *rateField;
    IBOutlet NSTextField *totalField;
}
@end


@implementation ConverterController
- (IBAction) convert: (id) sender {
    float amount = [amountField floatValue];
    float rate = [rateField floatValue];
    float total = [converter
        convertAmount:amount atRate:rate];
    [totalField setFloatValue:total];
}
@end
```

# AppleScriptObjC

## Currency Converter

```
@interface ConverterController : NSObject {


    IBOutlet NSTextField *amountField;
    IBOutlet NSTextField *rateField;
    IBOutlet NSTextField *totalField;
}
@end


@implementation ConverterController
- (IBAction) convert: (id) sender {
    float amount = [amountField floatValue];
    float rate = [rateField floatValue];
    float total = [converter
        convertAmount:amount atRate:rate];
    [totalField setFloatValue:total];
}
@end
```

```
script ConverterController
  property parent: class "NSObject"









end script
```

# AppleScriptObjC

## Currency Converter

```
@interface ConverterController : NSObject {


    IBOutlet NSTextField *amountField;
    IBOutlet NSTextField *rateField;
    IBOutlet NSTextField *totalField;
}
@end


@implementation ConverterController
- (IBAction) convert: (id) sender {
    float amount = [amountField floatValue];
    float rate = [rateField floatValue];
    float total = [converter
        convertAmount:amount atRate:rate];
    [totalField setFloatValue:total];
}
@end
```

```
script ConverterController
  property parent: class "NSObject"

  property amountField: missing value
  property rateField: missing value
  property totalField: missing value









end script
```

# AppleScriptObjC

## Currency Converter

```
@interface ConverterController : NSObject {


    IBOutlet NSTextField *amountField;
    IBOutlet NSTextField *rateField;
    IBOutlet NSTextField *totalField;
}
@end


@implementation ConverterController
- (IBAction) convert: (id) sender {
    float amount = [amountField floatValue];
    float rate = [rateField floatValue];
    float total = [converter
        convertAmount:amount atRate:rate];
    [totalField setFloatValue:total];
}
@end
```

```
script ConverterController
  property parent: class "NSObject"

  property amountField: missing value
  property rateField: missing value
  property totalField: missing value



  on convert_(sender)



  end convert_
end script
```

# AppleScriptObjC

## Currency Converter

```
@interface ConverterController : NSObject {


    IBOutlet NSTextField *amountField;
    IBOutlet NSTextField *rateField;
    IBOutlet NSTextField *totalField;
}
@end


@implementation ConverterController
- (IBAction) convert: (id) sender {
    float amount = [amountField floatValue];
    float rate = [rateField floatValue];
    float total = [converter
        convertAmount:amount atRate:rate];
    [totalField setFloatValue:total];
}
@end
```

```
script ConverterController
  property parent: class "NSObject"

  property amountField: missing value
  property rateField: missing value
  property totalField: missing value



  on convert_(sender)
    set amount to amountField's floatValue()



  end convert_
end script
```

# AppleScriptObjC

## Currency Converter

```
@interface ConverterController : NSObject {


    IBOutlet NSTextField *amountField;
    IBOutlet NSTextField *rateField;
    IBOutlet NSTextField *totalField;
}
@end


@implementation ConverterController
- (IBAction) convert: (id) sender {
    float amount = [amountField floatValue];
    float rate = [rateField floatValue];
    float total = [converter
        convertAmount:amount atRate:rate];
    [totalField setFloatValue:total];
}
@end
```

```
script ConverterController
  property parent: class "NSObject"

  property amountField: missing value
  property rateField: missing value
  property totalField: missing value




  on convert_(sender)
    set amount to amountField's floatValue()
    set rate to rateField's floatValue()



  end convert_
end script
```

# AppleScriptObjC

## Currency Converter

```
@interface ConverterController : NSObject {


    IBOutlet NSTextField *amountField;
    IBOutlet NSTextField *rateField;
    IBOutlet NSTextField *totalField;
}
@end


@implementation ConverterController
- (IBAction) convert: (id) sender {
    float amount = [amountField floatValue];
    float rate = [rateField floatValue];
    float total = [converter
        convertAmount:amount atRate:rate];
    [totalField setFloatValue:total];
}
@end
```

```
script ConverterController
  property parent: class "NSObject"

  property amountField: missing value
  property rateField: missing value
  property totalField: missing value


  on convert_(sender)
    set amount to amountField's floatValue()
    set rate to rateField's floatValue()
    set total to converter's
       convertAmount_atRate_(amount, rate)

  end convert_
end script
```

# AppleScriptObjC

## Currency Converter

```
@interface ConverterController : NSObject {


    IBOutlet NSTextField *amountField;
    IBOutlet NSTextField *rateField;
    IBOutlet NSTextField *totalField;
}
@end


@implementation ConverterController
- (IBAction) convert: (id) sender {
    float amount = [amountField floatValue];
    float rate = [rateField floatValue];
    float total = [converter
        convertAmount:amount atRate:rate];
    [totalField setFloatValue:total];
}
@end
```

```
script ConverterController
  property parent: class "NSObject"

  property amountField: missing value
  property rateField: missing value
  property totalField: missing value

  on convert_(sender)
    set amount to amountField's floatValue()
    set rate to rateField's floatValue()
    set total to converter's
       convertAmount_atRate_(amount, rate)
    tell totalField to setFloatValue_(total)
  end convert_
end script
```

# AppleScriptObjC

## Currency Converter

```
@interface ConverterController : NSObject {


    IBOutlet NSTextField *amountField;
    IBOutlet NSTextField *rateField;
    IBOutlet NSTextField *totalField;
}
@end


@implementation ConverterController
- (IBAction) convert: (id) sender {
    float amount = [amountField floatValue];
    float rate = [rateField floatValue];
    float total = [converter
        convertAmount:amount atRate:rate];
    [totalField setFloatValue:total];
}
@end
```

```
script ConverterController
  property parent: class "NSObject"

  property amountField: missing value
  property rateField: missing value
  property totalField: missing value

  on convert_(sender)
    set amount to amountField's floatValue()
    set rate to rateField's floatValue()
    set total to converter's
       convertAmount_atRate_(amount, rate)
    tell totalField to setFloatValue_(total)
  end convert_
end script
```

# AppleScriptObjC

## Identifiers

- AppleScript terminology and reserved words must be escaped
  - Use |vertical bars| if necessary

```
script MyView
    property parent: class "NSView"

    on drawRect_(rect)
        NSColor's whiteColor()'s |set|()
        set r to my |bounds|()
    end drawRect_
end script
```

# AppleScriptObjC

## Classes and constants

- Treated as elements/properties of `current application`
  - `current application's class "NSObject"`
  - `current application's NSCaseInsensitiveSearch`

```
script MyView
    property parent: class "NSView"

    on drawRect_(rect)
        set c to current application's class "NSColor"'s whiteColor()
        c's colorUsingColorSpaceName_(
           current application's NSCalibratedRGBColorSpace)
    end drawRect_
end script
```

# AppleScriptObjC

## Classes and constants

- Treated as elements/properties of `current application`
  - `current application's class "NSObject"`
  - `current application's NSCaseInsensitiveSearch`

```
property NSColor: class "NSColor"
property NSCalibratedRGBColorSpace: my NSCalibratedRGBColorSpace

script MyView
    property parent: class "NSView"
    on drawRect_(rect)
        set c to NSColor's whiteColor()
        c's colorUsingColorSpaceName_(NSCalibratedRGBColorSpace)
    end drawRect_
end script
```

# AppleScriptObjC

## Referring to self

- Properties: `of me` or `my`

```
my radius
```

- Methods: `tell me to` or `my`

```
tell me to setNeedsDisplay_(true)
```

- Invoke inherited implementation: `continue`

```
continue init()
```

# AppleScriptObjC

## Data translation

- AppleScript types bridged to Foundation types automatically

```
-- string vs. NSString
aString's rangeOfString_("foobar")

-- list vs. NSArray
NSMutableArray's arrayWithArray_({1, 2, 3})

-- record vs. NSDictionary
NSDictionary's dictionaryWithDictionary_(
   {priority:1, dueDate: date "6/12/2009"})

-- record or list vs. struct
aString's substringWithRange({location:0, |length|:4})
aString's substringWithRange({0, 4})
```

# AppleScriptObjC

## Data translation

- Foundation types bridged to AppleScript types manually

```
on firstTenCharacters_(anNSString)
    set substring to anNSString's substringToIndex_(10)
    -- “substring” is an NSString, too.

    display dialog (substring as text)
    -- explicit coercion to “text”.
end
```

# AppleScriptObjC

## Data translation

- AppleScript idioms work on Cocoa objects

```
set docs to NSApp's orderedDocuments()
-- docs is an NSArray.

repeat with d in docs
    first item of d's readableTypes()
    -- “list” item access

    set n to displayName of d
    -- “property” access using KVC

    set d's displayName to (n as text) & "AppleScript"
    -- property setters, too!
end
```

# AppleScriptObjC

## NSError** parameters

- When calling
  - `missing value` for nil

```
tell fileManager to removeItemAtPath_error_(path, missing value)
```

  - `reference` to return the value

```
tell fileManager to removeItemAtPath_error_(path, reference)
--> {false, anNSError}
```

# AppleScriptObjC

## NSError** parameters

- When being called
  - `contents` of magic “pointer” parameter

```
on readFromURL_ofType_error_(url, type, outError)
    if outError is not missing value
        set theError to NSError's errorWithDomain_code_userInfo_(...)
        set contents of outError to theError
    end
    return false
end
```

# Demo

## AppleScriptObjC

**Michael Gorbach**

Just like “Горбачёв”, but without the “ёв”

# The Fine Print

Void where prohibited. Some assembly required. Action figures not included. This is not an offer to sell securities. Any resemblance to real persons, living or dead, is purely coincidental. No animals were harmed in the testing of this product. Contents may settle during shipment. Use only with proper ventilation. Keep away from open flames and avoid inhaling fumes. Do not use while operating a motor vehicle or heavy equipment. Apply only to affected area; if condition persists, consult your physician. May be too intense for some viewers. Post office will not deliver without postage. Best if used before date on carton. Times are approximate. Slightly enlarged to show detail. Colors may fade over time. Content has been edited for your screen. Not affiliated with the American Red Cross. At participating locations only.

# AppleScriptObjC

## AppleScript + Cocoa = win

- Access to all of Cocoa
  - Convenience features like bindings
  - Power features like CalendarStore
  - New features like…?
- Still regular AppleScript, too
  - Control other applications
  - Use existing AppleScript code

# Summary

- AppleScript is future-compatible
  - 64-bit
  - GCD-safe
- Write Cocoa applications
  - Use the language you like
  - Consistent experience across all languages
  - AppleScript Studio is deprecated
  - Use AppleScriptObjC

# More Information

**Matt Drance**
Framework Technologies Evangelist
mdrance@apple.com

**Sal Soghoian**
Automation Product Marketing
sal@apple.com

## Documentation, Sample Code, and Other Resources

http://developer.apple.com/wwdc
http://developer.apple.com/snowleopard/library/referencelibrary

# Labs

| | |
|---|---|
| AppleScript and Cocoa Bridge Lab | Mac OS X Lab C<br>Friday 2:00PM |

# Q&A

## WWDC Session Survey

You can rate your sessions on the WWDC Attendee Site.

Survey forms are located on each session description page at:

developer.apple.com/wwdc/attendee

WWDC09